WIRELESS COMMUNICATIONS AND MOBILE COMPUTING
Wirel. Commun. Mob. Comput. 2003; **3**:000–000 (DOI: 10.1002/wcm.133)

Design and performance analysis of leader election and initialization protocols on ad hoc networks

Chih-Shun Hsu and Jang-Ping Sheu*,†

Department of Computer Science and
Information Engineering
National Central University
Chung-Li,
32054
Taiwan

Summary

Leader election and initialization are two fundamental problems in mobile ad hoc networks (MANETs). The leader can serve as a coordinator in the MANETs and the initialization protocol can assign each host a sequential, unique, and short ID●. As we know, no● research on initialization for IEEE 802.11-based MANETs has been done. Here, we propose two contention-based leader election and initialization protocols for IEEE 802.11-based single-hop MANETs. We also provide an efficient approach to evaluate the performance of the proposed protocols, such that the performance can be evaluated in polynomial time. The evaluation results provide a guideline to set the size of the contention window and thus improve the performance of the proposed leader election and initialization protocols. The results can also be used as a guideline to set the size of the contention window for any contention-based protocol. Simulation results justify that the evaluation results provide a good guideline to set the size of the contention window. Copyright © 2003 John Wiley & Sons, Ltd.

KEY WORDS
initialization
leader election
mobile ad hoc network (MANET)

---

*Correspondence to: Jang-Ping Sheu, Department of Computer Science and Information Engineering, National Central University, Chung-Li, 32054, Taiwan.

†E-mail: sheujp@csie.ncu.edu.tw

Contract/grant sponsor: MOE Program for promoting Academic Excellent● of Universities; contract/grant number: A-91-H-FA07-1-4.

1. Introduction

Leader election and initialization have been studied extensively in traditional distributed and parallel systems [1–10]; they are also important in wireless networks. The leader is the coordinator of the network; it can serve as a relay point or it can coordinate its members' actions in networks. The initialization protocol can provide each host a sequential, unique, and short ID, so that each host can perform ID-based algorithms [11–16]. There are several leader election and initialization researches that have been done Q4 for wireless networks [16–26]. To the best of our• knowledge, none of them are designed to initialize an IEEE 802.11-based single-hop ad hoc network (MANETs).

A simple leader election algorithm for wireless LAN is proposed in [17]. The leader, which is elected by the base station, serves as a reporter to its multicast group members. It will send a feedback to the sender when there is no collision, and thus increase the reliability of the multicast. With a similar idea, a random leader-based reliable multicast protocol is proposed in [18], which overcomes the problem of feedback collision. Both the algorithms [17,18] based on wireless LAN require the help of the base station. Q5 Two leader election algorithms based on• TORA [27] for MANET are proposed in [19]. One algorithm is for a single topology change, and the other tolerates multiple topology changes. Both algorithms work by assigning each host a unique height (6-tuple), which is costly. A uniform leader election protocol for radio networks is proposed in [25]. Randomized leader election and initialization protocols for time-slotted single-hop MANETs are proposed in [20]. These protocols (termed as the *Nakano–Olariu* protocols) are efficient but based on an impractical assumption (termed as the *Nakano–Olariu* assumption) that the sender can detect its own transmission status. With a similar approach, an energy-efficient initialization protocol for wireless sensor networks is proposed in [22], and for single-hop radio networks is proposed in [21]. A leader election algorithm revised from [20] is proposed in [16]. In [16], the leader acts as a coordinator, which initializes the hosts of the same priority by giving each of them a unique ID, so that these hosts know when to transmit their frames according to their IDs. A hybrid randomized Q6 initialization protocol for• TDMA-based single-hop wireless networks is proposed in [24].

Q7 Since all the previous works• are either too costly [19] or cannot work properly in IEEE 802.11-based MANETs, we propose efficient leader election and initialization protocols for IEEE 802.11-based single-hop MANETs with and without the knowledge of the number of hosts. The proposed initialization protocols work as follows. First, elect a leader in the MANET, then let the leader serve as a detector, which will tell the sender the status of the transmission. If the transmission is successful, the leader will assign a unique and short ID to the sender. We set the size of contention window according to the number of hosts. When the number of hosts is not available, we propose a new adaptive round transmission protocol [28] to initialize the MANET. We set the value of contention window (CW) to a predetermined number. After a round, we can estimate the number of the hosts in the MANET according to the previous round's transmission status. In the next round, we can set the size of contention window according to the estimated number of hosts.

We also derive some recursive forms to evaluate the performance of the proposed initialization protocols. These recursive forms can be calculated in polynomial time. With the evaluation results, we can set the proper size of contention window so that the proposed protocols will take less time to elect a leader or initialize a MANET. The evaluation results can also be used as a guideline to set the contention window in any contention-based protocol. Simulation results justify that the evaluation results provide a good guideline to set the size of the contention window. Our protocols are not only more practical than the Nakano–Olariu protocols but they also perform better when they are based on the same assumption.

The rest of the paper is organized as follows. Preliminaries are given in Section 2. Section 3 presents our leader election and initialization protocols. Section 4 presents the numerical evaluation results for the proposed protocols. Simulation results are presented in Section 5. Section 6 concludes the paper.

2. Preliminaries

2.1. Assumptions

In this paper, we intend to solve the leader election and initialization problems on an IEEE 802.11-based single-hop MANET. We assume that every mobile host in the same MANET is synchronized and can detect the status of its neighboring host's transmission.

When the leader has been elected and the initialization procedure is over, the leader will broadcast a beacon periodically, so that other hosts can realize the existence of the leader and synchronize with the leader. Since each node can roam freely in the MANET, the leader or some nodes may leave the MANET. When any host does not hear the beacon sent by the leader for a certain period of time, the host may start a new leader election. When a new host joins the MANET, it will wait until it hears the beacon sent by the leader and then acquires an ID from the leader. When there is more than one leader in the MANET, the host with larger medium access (MAC) address will withdraw.

2.2. IEEE 802.11 MAC Protocol

The IEEE 802.11 medium access (MAC) protocol [29] used in MANETs is the distributed coordination function (DCF) that is based on the *Carrier Sense Multiple Access with Collision Avoidance* (CSMA/CA) mechanism. When a mobile host wants to transmit frames, it first detects the status of the medium. If the medium is busy, the host will defer until the medium is idle for a period of time equal to DCF interframe space (DIFS) . After this DIFS idle time, the host will generate a random backoff period, where $backoff\ time = Random() \times ST$. *Random* () is a random function, which is uniformly distributed between the interval [0, *CW*] and *ST* is the length of a backoff time slot. The initial value of the *CW* is $CW_{\min}$. When a host wants to send data, it first senses the medium. If the medium is idle for a period of time equal to DIFS, the backoff procedure will decrease the backoff time; otherwise, it will stop decreasing the backoff time. When the backoff timer expires, the host will transmit the frame. After the sender transmits the frame, if it is a broadcast, the receivers do nothing. Otherwise, if it is a unicast, the receiver will wait for a period of time equal to short interframe space (SIFS, $SIFS < DIFS$) and then reply an Ack● to the sender. If the sender does not receive an Ack from the receiver, the sender will double the size of its contention window and repeat the DCF procedure again.

2.3. The Nakano–Olariu Protocols

The Nakano–Olariu protocols [20] are based on a time-slotted single-hop MANET. They assume that the mobile host can detect the status of its own transmission. If the mobile host has the collision detection capability, it can detect three status, namely, NULL (no transmission), SINGLE (exactly one transmission), and COLLISION (two or more transmissions), of the radio channel at the end of a time slot. However, when the mobile host has no collision detection capability, it can only detect two status, namely, SINGLE (exactly one transmission) and NOISE (collision or no transmission). Under this condition, the mobile hosts in the MANET need to elect a leader to help them to distinguish NULL from COLLISION.

When the mobile host has no collision detection capability and the number of mobile hosts in the MANET is unknown in advance, each mobile host contends to be the leader. At first, each mobile host transmits with probability $1/2$. If the status of the channel is SINGLE, the mobile host that has transmitted in the previous time slot is declared as the leader. Otherwise, each mobile host continues to transmit with half of the previous probability until a host is declared as the leader.

When the mobile host has the collision detection capability and the number of mobile hosts in the MANET (denoted as n) is known in advance, the mobile hosts need not elect a leader; they get their IDs by contention. At first, each mobile host transmits with probability $1/m$, where m is the number of hosts that have no ID. If the channel status is SINGLE, the mobile host that has transmitted in the previous time slot gets $n - m + 1$ as its ID. The other hosts that have no ID will follow the same procedure to get their IDs until there is no host without ID.

If n is unknown in advance, each mobile host will follow the idea of the partition tree to get its own ID. The partition tree is a binary tree that each host flips a fair coin to decide which subtree it belongs to. In the beginning, all the hosts belong to the root nodes of the tree and all the hosts transmit on the channel simultaneously. Since the channel status is COLLISION, each host flips a fair coin to partition the tree. The host, which flips 'head', is assigned to the left subtree; otherwise, it is assigned to the right subtree. Only the host assigned to the left subtree can transmit on the channel in the next time slot. The left subtree will be recursively partitioned until the subtree contains only one host. When only one host transmits on the channel, the channel status is SINGLE and thus the host can get its own ID. The host in the left-most leaf of the tree will get its ID first and then, in the same manner, the hosts in other subtrees will recursively partition the subtrees and get their IDs until all the hosts have got their IDs.

In the Nakano–Olariu protocols,● irrespective of whether the mobile host has the collision detection capability or not, the assumptions are not practical. When the mobile host is transmitting message, it is very hard for itself to detect the channel status. Therefore, we propose more practical leader election and initialization protocols, which are based on the standard of IEEE 802.11 protocol.

3. Our Leader Election and Initialization Protocols

Two efficient leader election and initialization protocols for IEEE 802.11-based single-hop MANETs are proposed herein—one is for a MANET whose number of hosts is known in advance, and the other one is for a MANET without the knowledge of the number of hosts. In the following text, we assume that the number of hosts in the MANET is known in advance.

3.1. The Leader Election Protocol

Before initializing a MANET, we need to elect a leader to serve as a coordinator in the network. Every host in the network has an equal chance of becoming a leader. Without loss of generality, we assume that there are n hosts, H_1, H_2, H_3, $\ldots$, H_n, in the MANET. In the beginning, every host basically follows a DCF procedure, mentioned in Section 2.2, to contend as a leader in the MANET. However, the value of CW is set according to the number of hosts in the MANET, and each host will not contend again until an election round is over (CW is set as $m-1$ and an election round is said to be over if the $(m-1)$th backoff time slot has expired). When host H_i's $(i = 1 \ldots n)$ backoff timer has expired and there is no other host that has successfully claimed itself as the leader, host H_i claims itself as the leader by broadcasting its MAC address. Assume that host H_a successfully broadcasts its MAC address. Since there is no collision, the claim can be heard by all hosts except H_a. Once the claim is successful, the other hosts, whose backoff timers have not expired, will wait until their backoff timers have expired and then send an acknowledgement to H_a. They broadcast H_a's MAC address to inform H_a that it is the new leader of the MANET. Again, if there is only one host, say H_b, broadcasting H_a's MAC address, the acknowledgement can be heard by all hosts except H_b, the acknowledgement is said to be successful and all the hosts except H_b know that a new leader has been elected. After a short period of time equal to SIFS, host H_a can announce itself as the new leader and the leader election process is successful and completed.

According to the above description, a successful leader election process requires at least two successful broadcasts from different hosts (a broadcast is said to be successful if there is only one host broadcasting the message). It first requires a successful claim, and then a successful acknowledgement from another host. If an election round is over and no host is elected as a leader, every host follows the same procedure to contend as the leader in the next round until a new leader has been elected. The size of CW is set to $m-1$ in each election round. If there is a host, say H_a, whose claim is successful but the acknowledgements sent by other hosts are all failed after an election round, all the hosts except H_a in the next election round will inform host H_a until the acknowledgement is successful. Note that H_a will still broadcast a claim message because H_a does not know that it has a successful claim.

For example, assume that there are eight hosts A, B, C, D, E, F, G, and H, in the MANET and the length of a backoff time slot is denoted as *ST*. In the beginning, each host set its backoff timer as $Random() \times ST$, where *Random*() is uniformly distributed between the interval [0,7]. As Figure 1 shows, the backoff timers of hosts B and C are set as 0, host F is set as $2 \times ST$, hosts A and E are set as $3 \times ST$, hosts D and H are set as $4 \times ST$, and host G is set as $6 \times ST$, respectively. After a DIFS, hosts B and C claim themselves as the leaders of the MANET and a collision occurs, so this claim is not successful. After host F's backoff timer has expired, host F claims itself as the leader. Since there is only one host claiming itself as the leader, the claim is successful. Therefore, hosts A, D, E, H, and G stop claiming themselves as the leaders; they all try to send an acknowledgement to host F by broadcasting its MAC address when their backoff timers have expired. Hosts A and E send their acknowledgements simultaneously and a collision occurs. The same thing happens as the backoff timers of hosts D and H have expired. Finally, host G's backoff timer expires, host G sends

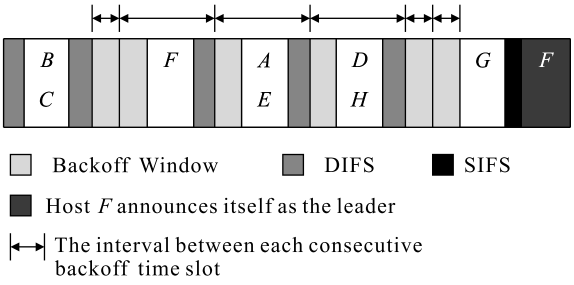

Fig. 1. An example of a successful leader election.

an acknowledgement and the acknowledgement is successful. After receiving the acknowledgement, the new leader, host *F*, waits for an SIFS and announces itself as the new leader of the MANET.

The leader election algorithm is shown as follows:

```
**Algorithm 1:** Leader-Election($n$,$m$)
$n$: number of hosts in the MANET
$m-1$: the value of CW
$H_a$: the first host broadcast its
  claim successfully
**Initial:** $Claim = false$, and every
  host randomly set its backoff
  timer as $R \times ST$, where $R \in N$ and
  $0 \le R \le m-1$.
**while** (no host announces that it is
  the leader) **do**
 **if** the ($m-1$)-th backoff time
   slot has expired **then** every host
   randomly set its backoff timer as
   $R \times ST$.
 When any host $H_i$'s backoff timer
   expires $\Rightarrow$
  **if** ($Claim = false$) **then** host $H_i$
   broadcasts its own MAC address.
  **else** host $H_i$ broadcasts host $H_a$'s
   MAC address.
  **endif**
 **if** a host can hear a successful
   broadcast from $H_i$ **then**
  **if** (the received MAC address =
    my MAC address) **then** the host
    waits for a period of time equal
    to SIFS and then announces
    itself as the leader.
  **else** $Claim = true$ and $H_a = H_i$
  **endif**
 **endif**
**endwhile**
```

3.2. The Initialization Protocol

After the leader has been elected, we can initialize the MANET with the help of the leader. Every host (except the leader) sets the value of CW as $m-1$ according to the number of hosts and basically follows the DCF procedure to send a request ID message by broadcasting its own MAC address. If the leader can receive the request ID message without any collision, it will assign an ID to the host by broadcasting the host's ID after receiving the request ID message for a period of time equal to SIFS. When the $(m-1)$th backoff time slot has expired and all the hosts (except the leader) have broadcast their request ID messages, an initialization round is over. Assume that before the initialization round begins, $r1$ hosts have not being assigned IDs by the leader and after the initialization round is over, there are still $r2$ hosts with no ID. In the next initialization round, the hosts with no ID will reset the value of CW as $m-1$, where $m = \lceil m \times r2/r1 \rceil$. The initialization procedure will repeatedly be executed until all the hosts have got their IDs.

For example, assume that there are four hosts in the MANET. Host *A* is the elected leader in the MANET with $ID = 1$. First, the other three hosts (*B*, *C*, *D*) set the initial value of CW as 2, and then follow the DCF procedure to send their request ID messages. As Figure 2 shows, host *C* sets its backoff timer as 0, hosts *B* and *D* set their backoff timers as $2 \times ST$. So host *A* (the leader) can receive host *C*'s request ID message without any collision. After a SIFS, host *A* broadcasts host *C*'s ID(=2). When hosts *B* and *D*'s backoff timers expire, they broadcast their request ID messages simultaneously and a collision occurs, so host *A* cannot receive the request ID message successfully. When the first initialization round is over, hosts *B* and *D* both change the value of CW to 1 and follow the DCF procedure to send their request ID messages. In the second initialization round, host *B* sets its backoff timer as *ST* and host *D* sets its backoff timer as 0. This time, host *A* can receive the request ID messages of both hosts *D* and *B* successfully, so host *A* assigns 3 and 4 to hosts *D* and *B*, respectively. Finally, every host in the MANET has its own ID and complete● the initialization procedure. Q10

The initialization algorithm is shown as follows:

```
**Algorithm 2:** Initialization($n$, $m$)
$n$: number of hosts in the MANET
$m-1$: the value of CW
```

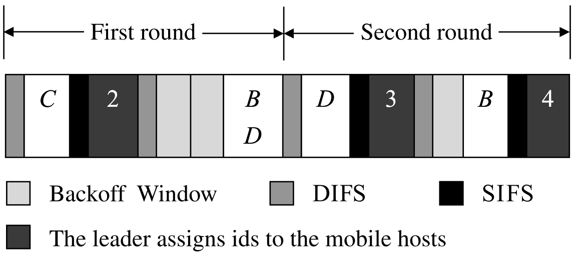

Fig. 2. An example of the initialization protocol.

```
$r1$: number of hosts that have not got their IDs before an initialization round
$r2$: number of hosts that have not got their IDs after an initialization round
**Initial:** $id = 1$, $r1 = r2 = n - 1$ and every host (except the leader) randomly set its backoff timer as $R \times ST$, where $R \in N$ and $0 \leq R \leq m - 1$.
**while** $(id \neq n)$ **do**
  **if** the $(m-1)$th backoff time slot has expired **then**
    $m = \lceil m \times r2/r1 \rceil$, $r1 = r2$
    The hosts that have not obtained their IDs randomly set their backoff timers as $R \times ST$.
  **endif**
  When any host $H'$s backoff timer expires $\Rightarrow$
    Host $H$ broadcast a request ID message.
    **if** the leader detects that there is no collision **then**
      $id = id + 1$
      The leader waits for a period of time equal to SIFS and then assigns ID to host $H$ by broadcasting $H'$s ID.
      The hosts with no ID set $r2 = r2 - 1$.
    **endif**
**endwhile**
```

4. Performance Evaluation of Our Protocols

We proposed several efficient algorithms to evaluate the performances of our protocols herein. With the evaluation results, we can decide how to set the proper initial value of CW so that the proposed protocols will take less time to elect a leader and initialize a MANET. The results can also be used as a guideline to set the initial value of CW in any IEEE 802.11-based MANET as long as the number of potential contenders is known in advance. For the ease of understanding, we first evaluate the average number of time slots required for the proposed protocols based on a time-slotted MANET. We then make a more accurate evaluation based on the IEEE 802.11 standard.

4.1. Evaluation Based on a Time-slotted MANET

4.1.1. Performance evaluation of the leader election protocol

Before our performance evaluation, we define the following notations:

- $p(n, m, 0)$: probability with no successful broadcast in an election round.
- $p(n, m, 1)$: probability with only one successful broadcast in an election round.
- $p(n, m, 2^+)$: probability with at least two successful broadcasts in an election round.
- $s(n, m, 1)$: the expected number of time slots required to detect the first successful broadcast under the condition of at least two successful broadcasts in an election round.
- $s(n, m, 2)$: the expected number of time slots required to detect the second successful broadcast under the condition of at least two successful broadcasts in an election round.

Figure 3 shows the state transition diagram of the leader election protocol. By definition, the probability that the state transits from 'Start Election' to 'Successful Ack' is $p(n, m, 2^+)$, from 'Start Election' to 'Successful Claim' is $p(n, m, 1)$, and remains in state 'Start Election' is $p(n, m, 0)$. Once the state transits to 'Successful Claim', it needs at least one more successful broadcast. The successful broadcast should not be transmitted by the host, H_a, which makes the successful claim in the previous election round. Because H_a does not know that it has a successful claim in the previous election round, its successful broadcast in the next election round will not become a successful acknowledgement, so the

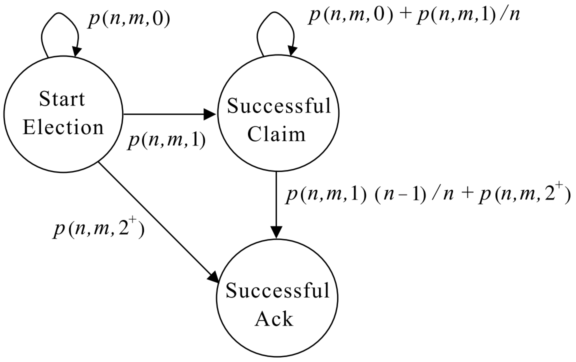

Fig. 3. A state transition diagram of our leader election protocol.

state will remain in 'Successful Claim'. The probability of this case is $p(n, m, 1)/n$. If there is no successful broadcast in the next election round, the state also remains in 'Successful Claim'. The probability of this case is $p(n, m, 0)$. The total probability that the state remains in 'Successful Claim' is $p(n, m, 0) + p(n, m, 1)/n$ and the probability that the state transits from 'Successful Claim' to 'Successful Ack' is $p(n, m, 1)(n-1)/n + p(n, m, 2^+)$.

Figure 4 shows the average number of time slots required for each state transition. By definition, the average number of time slots required for the state transition directly from 'Start Election' to 'Successful Ack' is $s(n, m, 2)$. The average number of time slots required for the state transition from 'Start Election' to 'Successful Claim' is m, since another election round is required to transit the state from 'Successful Claim' to 'Successful Ack'. The state transition from 'Successful Claim' to 'Successful Ack' requires at least one more successful broadcast in an election round. If there is only one successful broadcast in the election round, the average number of time slots required is $m + 1/2$, because the chance that the only one successful broadcast occurs in time slot 1, 2, …, or m is equal, the average number of time slots required in this case is $\frac{\sum_{k=1}^{m} k}{m} = m + 1/2$. If there are more than one successful broadcast in the election round, by definition, the average number of time slots required for the first successful broadcast is $s(n, m, 1)$. If the state remains in 'Start Election' or 'Successful Claim', it will require another election round, whose length is m time slots, to transit the state to 'Successful Ack'.

According to Figures 3 and 4, as long as we calculate $p(m, n, 0)$, $p(m, n, 1)$, $p(m, n, 2^+)$, $s(n, m, 1)$, and $s(n, m, 2)$, we can obtain the expected number of time slots required for a successful leader election. Although we can evaluate the successful probabilities and the expected number of time slots required for our leader election protocol by generating all possible combinations and then get the final results, yet, analysis by the brute force approach is too costly and the time complexity is $O(m^n)$. Therefore, we derive several recursive forms to calculate the successful probability and the expected time slots needed for our leader election protocol.

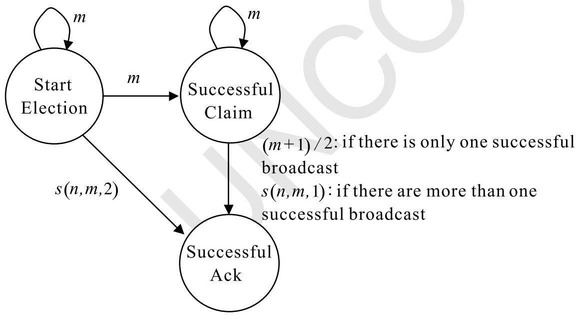

Fig. 4. The average number of time slots required for each state transition.

For the convenience of calculating $p(m,n,0)$, $p(m,n,1)$, $p(m,n,2^+)$, $s(n,m,1)$, and $s(n,m,2)$, we first calculate the probability $pb(k)$, where $pb(k)$ is the probability that k ($k = 0, 1, \ldots, n$) hosts broadcast their messages in the mth time slot and the other $n-k$ hosts broadcast their initialization messages in the first $(m-1)$ time slots. Since each host has the same probability to broadcast its message in any of the m time slots, the probability that a host broadcast in the mth time slot is $1/m$ and the probability that a host broadcast in the first $m-1$ time slots is $m - 1/m$. The number of combinations that randomly choose k hosts from n hosts is $C_k^n = n!/(n-k)!k!$. After analysis, we have $pb(k) = C_k^n (1/m)^k (m - 1/m)^{n-k}$.

First, consider $p(n,m,0)$ with the following two cases:

- **Case 1:** Assume that there is no host broadcasting in the mth time slot and no host makes any successful broadcast in a round. The probability in this case is $pb(0)p(n, m-1, 0)$.
- **Case 2:** Assume that there are k hosts broadcasting in the mth time slot, where $k = 2, \ldots, n$.($k \neq 1$, if $k = 1$ there will be a successful broadcast.) In other words, $(n-k)$ hosts broadcasting in the first $m-1$ time slots and no host makes any successful broadcast in a round. The probability for each k is $pb(k)\ p(n-k,\ m-1,\ 0)$.

The sum of the above two cases is the probability with no successful broadcast in an election round. We have $p(n, m, 0) = pb(0)p(n, m-1, 0) + \sum_{k=2}^{n} pb(k)p(n-k, m-1, 0)$.

We can also derive the recursive forms of $p(n,m,1)$ and $p(n,m,2^+)$ in the similar manner. We have $p(n, m, 1) = pb(0)p(n, m-1, 1) + pb(1)p(n-1, m-1, 0) + \sum_{k=2}^{n-1} pb(k)p(n-k, m-1, 1)$ and $p(n, m, 2^+) = pb(1)p(n-1, m-1, 1) + \sum_{k=0}^{n-2} pb(k)p(n-k, m-1, 2^+)$.

For $n \geq 3$ and $m \leq 2$, we can directly derive the following forms:

$$p(n, 1, 0) = 1,\ p(n, 1, 1) = 0 \text{ and } p(n, 1, 2^+) = 0$$
$$p(n, 2, 0) = 2^n - 2n/2^n,\quad p(n, 2, 1) = 2n/2^n \quad \text{and}$$
$$p(n, 2, 2^+) = 0$$

For $n \geq 4$ and $m \geq 3$, we use the above forms as basis, and then use the recursive forms to calculate $p(n, m, 0)$, $p(n, m, 1)$, and $p(n, m, 2^+)$ by applying the following algorithm.

```
Algorithm 3: Successful-Probability
  (n,m)
for $j = 3$ to $m$ do
  for $i = 4$ to $n$ do
    $p(i, j, 0) = pb(0)\, p(i, j-1, 0)$
      $+ \sum_{k=2}^{i} p(k)\ p(i-k, j-1, 0)$
    $p(i, j, 1) = pb(1)\, p(i-1, j-1, 0)$
      $+ pb(0)\, p(i, j-1, 1)$
      $+ \sum_{k=2}^{i-1} pb(k)\ p(i-k, j-1, 1)$
    $p(i, j, 2^+) = pb(1)\, p(i-1, j-1, 1)$
      $+ \sum_{k=0}^{i-2} p(k)\ p(i-k, j-1, 2^+)$
  endfor
endfor
```

Table I shows the numerical analysis results of $p(n, m, 2^+)$. As we can see when $m \geq 32$, the successful probability is close to 100%, for $20 \leq n \leq 100$. If the contention window size is large enough, the leader can be elected in an election round.

As shown in Figure 4, we need to calculate $s(n, m, 1)$ and $s(n, m, 2)$ before we calculate the expected number of time slots required for a successful leader election. We can derive the recursive forms of $s(n, m, 1)$ and $s(n, m, 2)$ and then calculate them in polynomial time. First, consider $s(n, m, 1)$ with the following two cases:

- **Case 1:** Assume that the second successful broadcast occurs in the mth time slot, which indicates that the first successful broadcast occurs in the first $(m-1)$ slots. The probability in this case is $pb(1)p(n-1, m-1, 1)/p(n, m, 2^+)$. The first successful broadcast can occur in any of the first $(m-1)$ time slots and the chance is equal for each time slot. Therefore, the average number of time slots required for the first successful broadcast in this case is $\frac{\sum_{k=1}^{m-1} k}{m-1} = m/2$.

Table I. Numerical evaluation results of $p(n, m, 2^+)$ (there are at least two successful broadcasts in an election round).

$n \backslash m$	8	16	32	64	128	256
20	0.5162	0.9934	0.9999	1.0000	1.0000	1.0000
40	0.0118	0.8965	0.9999	1.0000	1.0000	1.0000
60	0.0001	0.3986	0.9998	1.0000	1.0000	1.0000
80	0.0000	0.0744	0.9971	1.0000	1.0000	1.0000
100	0.0000	0.0092	0.9616	1.0000	1.0000	1.0000

- **Case 2:** Assume that there are at least two successful broadcasts occurring in the first $(m-1)$ time slots and there are k hosts broadcast in the mth time slot, $(n-k)$ hosts broadcast in the first $(m-1)$ time slots, where $k = 0 \ldots n-2$. The probability for each k is $pb(k)p(n-k, m-1, 2^+)/p(n, m, 2^+)$. Since the first successful broadcast occurs in the first $(m-1)$ time slots and there are at least two successful broadcasts in the first $(m-1)$ time slots, the average number of time slots required for the first successful broadcast in this case is $s(n-k, m-1, 1)$.

According to the above analysis, we can derive a form to calculate the expected value of $s(n, m, 1)$, which is

$$\frac{pb(1)p(n-1, m-1, 1)m/2 + \sum_{k=0}^{n-2} pb(k)p(n-k, m-1, 2^+)s(n-k, m-1, 1)}{p(n, m, 2^+)}$$

In a similar manner, we can derive the recursive form of $s(n, m, 2)$, which is

$$\frac{pb(1)p(n-1, m-1, 1)m + \sum_{k=0}^{n-2} pb(k)\, p(n-k, m-1, 2^+)s(n-k, m-1, 2)}{p(n, m, 2^+)}$$

Combining the results in Figures 3 and 4, we can use $p(m, n, 0)$, $p(m, n, 1)$, $p(m, n, 2^+)$, $s(n, m, 1)$, and $s(n, m, 2)$ to calculate the expected value (denoted as $ES(n, m)$) of the average number of time slots required for a successful leader election :

- **Case 1:** The state transits from 'Start Election' to 'Successful Claim', and then transits from 'Successful Claim' to 'Successful Ack'.

We first calculate the expected number of time slots required for the state transiting from 'Successful Claim' to 'Successful Ack' (denoted as $CA(n, m)$). When the state has transited to 'Successful Claim', it requires at least one more successful broadcast to transit to 'Successful Ack'. As Figures 3 and 4 show, if there is only one successful broadcast in the election round, the average number of time slots required in this case is $(m+1)/2$ and the probability is $p(n, m, 1)(n-1)/n$. If there are at least two successful broadcasts in the election round, the average number of time slots required for the first successful broadcast in this case is $s(n, m, 1)$ and the probability is $p(n, m, 2^+)$. If it remains in state

'Successful Claim' for k election rounds, it will take another km slots and the probability is $(p(n,m,0) + p(n,m,1)/n)^k$, where $p(n,m,0)$ is the probability that there is no successful broadcast in an election round and $p(n,m,1)/n$ is the probability that only the potential leader has a successful broadcast in an election round. We can calculate the expected value of $CA(n,m)$ as follows:

$$\sum_{k=0}^{\infty}(p(n,m,0) + p(n,m,1)/n)^k (p(n,m,1) \times (n-1)/n)((m+1)/2 + km) + \sum_{k=0}^{\infty}(p(n,m,0) + p(n,m,1)/n)^k p(n,m,2^+)(s(n,m,1) + km)$$

The $(p(n,m,0) + p(n,m,1)/n)^k (p(n,m,1)(n-1)/n)$ is the probability that the state remains in 'Successful Claim' for k election rounds and then gets only one successful broadcast in the next election round. The $(m+1)/2 + km$ is the average number of time slots required for this case. The $(p(n,m,0) + p(n,m,1)/n)^k p(n,m,2^+)$ is the probability that the state remains in 'Successful Claim' for k election rounds and then gets more than one successful broadcast in the next election round. $s(n,m,1) + km$ is the average number of time slots required for this case.

We then calculate the expected number of time slots required for the state transiting from 'Start Election' to 'Successful Claim' and from 'Successful Claim' to 'Successful Ack'. The expected value (denoted as $ES1(n,m)$) can be calculated as follows:

The state transits from 'Start Election' to 'Successful Claim' required m time slots and the state transit from 'Successful Claim' to 'Successful Ack' required $CA(n,m)$ time slots. Therefore, the state transits from 'Start Election' to 'Successful Claim' and then transits to 'Successful Ack' totally required $m + CA(n,m)$ slots. If the state remains in 'Start Election' for k election rounds, the average number of time slots becomes $(k+1)m + CA(n,m)$. The probability that the state transits from 'Start Election' to 'Successful Claim' is $p(n,m,1)$ and the probability that the state remains in 'Start Election' for k election rounds is $p(n,m,0)^k$. With the above results, we have

$$ES1(n,m) = \sum_{k=0}^{\infty} p(n,m,0)^k p(n,m,1)((k+1)m + CA(n,m)) = \frac{p(n,m,1)CA(n,m)}{1-p(n,m,0)} + \frac{p(n,m,1)m}{(1-p(n,m,0))^2}$$

- **Case 2:** The state transits directly from 'Start Election' to 'Successful Ack', or it remains in the 'Start Election' state k times and then transits directly to 'Successful Ack'. The average number of time slots required for the state transits directly from 'Start Election' to 'Successful Ack' is $s(n,m,2)$ and the probability is $p(n,m,2^+)$. If the state remains in 'Start Election' for k election rounds, it will take another km time slots and the probability is $p(n,m,0)^k$. Therefore, if the state remains in 'Start Election' for k election rounds and then transits directly to 'Successful Ack', the average number of time slots required for this case is $s(n,m,2) + km$ and the probability is $p(n,m,0)^k p(n,m,2^+)$. With the above results, we have

$$ES2(n,m) = \sum_{k=0}^{\infty} p(n,m,0)^k p(n,m,2^+)(s(n,m,2) + km) = \frac{p(n,m,2^+)s(n,m,2)}{1-p(n,m,0)} + \frac{p(n,m,0)m}{(1-p(n,m,0))^2}$$

The expected number of time slots required for a successful claim and acknowledgement is $ES1(n,m) + ES2(n,m)$. When the state has transited to 'Successful Ack', it needs another time slot for the new leader to announce itself as the leader, so the average number of time slots required for a successful leader election is $ES(n,m) = ES1(n,m) + ES2(n,m) + 1$.

In the following, we show the detailed algorithm to calculate the expected number of time slots:

```
Algorithm 4: Leader-Election-Time-
    Slots($n$,$m$)
Initial: $s(2,2,1) = 1$, $s(2,2,2) = 2$
Begin
Call Successful-Probability($n$,$m$)
for $i = 2$ to $n$ do
    for $j = 3$ to $m$ do
        $s(i,j,1) = \frac{\sum_{k=0}^{i-2} p(k)\, p(i-k, j-1, 2^+)\, s(i-k, j-1, 1) + pb(1)\, p(i-1, j-1, 1)\, j/2}{p(i, j, 2^+)}$
        $s(i,j,2) = \frac{\sum_{k=0}^{i-2} pb(k)\, p(i-k, j-1, 2^+)\, s(i-k, j-1, 2) + pb(1)\, p(i-1, j-1, 1)\, j}{p(i, j, 2^+)}$
    endfor
endfor
$CA(n,m) = \sum_{k=0}^{\infty}\left(p(n,m,0) + \frac{p(n,m,1)}{n}\right)^k$
    $\left(\frac{p(n,m,1)(n-1)}{n}\right)((m+1)/2 + km)$
```

```
  $+ \sum_{k=0}^{\infty} \left( p(n,m,0) + \frac{p(n,m,1)}{n} \right)^k$
  $p(n,m,2^+)\,(s(n,m,1) + km)$
$ES1(n,m) = \frac{p(n,m,1)\,CA(n,m)}{1 - p(n,m,0)}$
  $+ \frac{p(n,m,1)\,m}{(1 - p(n,m,0))^2}$
$ES2(n,m) = \frac{p(n,m,2^+)\,s(n,m,2)}{1 - p(n,m,0)}$
  $+ \frac{p(n,m,0)\,m}{(1 - p(n,m,0))^2}$
$ES(n,m) = ES1(n,m) + ES2(n,m) + 1$
End
```

The time complexity and space complexity of this algorithm are $O(mn^2)$ and $O(mn)$, respectively. Table II shows the numerical evaluation results of $ES(n,m)$. If n is fixed, when m approaches to n, the average number of time slots required for a successful leader election is optimal. The optimal number of time slots is about 6.5.

Table II. Numerical evaluation results of $ES(n,m)$ (the expected number of time slots required for a successful leader election).

$n\backslash m$	8	16	32	64	128	256
20	10.66	6.44	6.82	9.44	15.36	27.46
40	73.48	10.47	6.51	6.9	9.59	15.65
60	708.67	24.06	7.86	6.4	7.75	11.65
80	7674.4	65.29	10.56	6.55	6.94	9.65
100	88614	190.41	15.24	7.07	6.56	8.52

4.1.2. Performance evaluation of our initialization protocol

We assume that there are n hosts (except the leader) in the MANET and there are m slots in an initialization round. Under this condition, we want to know how many hosts (denoted as $NS(n,m)$) can get their IDs in an initialization round. According to this result, we can estimate the optimal number of time slots (denoted as $SSI(n)$) required for our initialization protocol.

To calculate the expected value of $NS(n,m)$, we need to calculate the probability and the average number of hosts that get their IDs in each case. We can derive the recursive form of $NS(n,m)$ according to the following analysis:

- **Case 1:** Assume that there are $(n-1)$ hosts that broadcast their initialization messages in the first $(m-1)$ time slots, and only one host broadcasts its initialization message in the mth time slot. The total number of hosts that get their IDs successfully in this case is $NS(n-1, m-1) + 1$ and the probability is $pb(1)$.
- **Case 2:** Assume that there are $(n-k)$ hosts that broadcast their initialization messages in the first $(m-1)$ time slots and k $(k \neq 1)$ hosts broadcast their initialization messages in the mth time slot. There will be $NS(n-k, m-1)$ hosts that get their IDs successfully and the probability for each k is $pb(k)$.

From the above analysis, we have

$$NS(n,m) = p(1)(NS(n-1, m-1) + 1) + \sum_{k=0, k\neq 1}^{n} p(k) NS(n-k, m-1)$$

The algorithm to calculate $NS(n,m)$ is as follows:

```
Algorithm 5: Number-of-Successful-Hosts($n$, $m$)
$NS(n,m)$: the expected number of hosts that get their IDs in an initialization round
for $i = 0$ to $n$ do
  for $j = 0$ to $m$ do
    if $(i = 0)$ or $(j = 0)$ then $NS(i,j) = 0$
    else if $(i = 1)$ then $NS(i,j) = 1$
    else if $(j = 1)$ and $(i \geq 2)$ then
      $NS(i,j) = 0$
    else $NS(i,j) = pb(1)(NS(i-1, j-1) + 1) + \sum_{k=0, k \neq 1}^{i} C(i,k)\, pb(k)\, NS(i-k, j-1)$
    endif
  endfor
endfor
```

Table III shows the expected number of hosts ($NS(n,m)$) that can get their IDs in an initialization round. As we can see, the expected number of hosts that• can get their IDs in an initialization round increasing with the number of time slots under the number of hosts is fixed.

Q12

Table III. Number of hosts ($NS(n,m)$) that can get their IDs in an initialization round.

n/m	8	16	32	64	128	256
20	1.5819	5.8679	10.941	14.828	17.231	18.567
40	0.219	3.228	11.596	21.643	29.459	34.337
60	0.0227	1.3319	9.2181	23.693	37.773	47.628
80	0.0021	0.4885	6.5135	23.056	43.052	58.723
100	0.00018	0.1679	4.3148	21.033	46.002	67.877

In fact, we can also use Algorithm 6 to estimate how many hosts will have successful transmission in a single-hop MANET when the number of hosts and the size of contention window are known in advance.

When a host successfully broadcasts its initialization message, it will take an extra time slot for the leader to broadcast the ID, so the average number of time slots required for a host to get its own ID is $m/[NS(n, m)] + 1$. Table IV shows the average number of time slots required for a host to get its own ID. The smaller the number, the more efficient is the initialization. When m is closest to n, we have the optimal result for our initialization and the expected number of time slots required for a host to get its own ID is about 3.7. After each initialization round, we pick a number (denoted as *OPT(r2)*) from 8, 16, 32, 64, 128, and 256, which is closest to $r2$, where $r2$ is the number of hosts that have not obtained their IDs in the end of every initialization round. We then set the size of the contention window as *OPT(r2)* to obtain the optimal result. When we get the result, we need to add n to this result, because every host needs an extra slot for the leader to broadcast the assigned ID.

The algorithm to calculate the average number of time slots required for initializing a MANET is shown as follows:

```
Algorithm 6: Initialization-Time(n)
SSI(n): number of time slots requi-
  red to initialize n hosts (except
  the leader)
r2: number of hosts that have not
  obtained their IDs
Initial: r2 = n, SSI(n) = 0
begin
While (r2 > 0)
  call Number-of-Successful-Hosts
    (r2, OPT(r2))
  Temp = NS(r2, OPT(r2))
  SSI(n) = SSI(n) + Temp
  r2 = r2 - Temp
endwhile
SSI(n) = SSI(n) + n
end
```

Table IV. Average number of time slots required for a host to get its own ID ($m/[NS(n, m)] + 1$).

$n\backslash m$	8	16	32	64	128	256
20	6.05	3.73	3.9248	5.32	8.43	14.79
40	37.54	5.96	3.76	3.96	5.35	8.46
60	352.94	13.01	4.47	3.7	4.39	6.38
80	3814.9	33.76	5.91	3.78	3.97	5.36
100	44086	96.27	8.42	4.04	3.78	4.77

4.2. Evaluation on an IEEE 802.11-based MANET

In Section 4.1, we evaluate the performance of the proposed protocols based on a time-slotted MANET, where each slot has an equal length. However, the length of each time slot (or the interval between each consecutive backoff time slot) is different in IEEE 802.11. As Figure 1 shows, when there are broadcast messages between the two consecutive backoff time slots, the interval is $ST + DIFS + SLOT$; otherwise, it is ST, where ST is the length of a backoff time slot and $SLOT$ is the length of time to transmit a leader election or initialization packet. The evaluation results in Section 4.1 are suitable for a time-slotted MANET, but are not suitable for an IEEE 802.11-based MANET. To make a more accurate evaluation for the proposed protocols on IEEE 802.11, we need to calculate the average length of each time slot (or the average interval between each consecutive backoff time slot). To calculate the average length of each time slot, we must evaluate how many time slots have been chosen to transmit messages. The transmissions in each chosen time slot can be a successful broadcast or a collision. Therefore, the sum of the average number of successful broadcasts and collisions in a round is the number of time slots that have been used to transmit message.

In Section 4.1, we propose a recursive form to calculate the number of hosts that have successful broadcasts (or get their IDs) in a round with m slots. To evaluate how many time slots have been chosen to transmit messages, we also need to evaluate the expected number of collisions in a round with m slots. In a similar manner, we can derive a recursive form to calculate the expected number of collisions (denoted as $COL(n, m)$) in a round with m slots. We have $COL(n, m) = pb(0)COL(n, m-1) + pb(1)COL(n-1, m-1) + \sum_{k=2}^{n} pb(k)(COL(n-k, m-1) + 1)$, where $COL(0, 0) = 0$, $COL(1, m) = 1$, when $m > 0$, and $COL(n, 1) = 0$, when $n > 1$.

Table V shows the average number of collisions in a round. When the size of the contention window is fixed, the average number of collisions increases as the number of hosts increases. Therefore, if the number of hosts is not known in advance, we can use an interpolation method to estimate how many hosts are in the MANET according to the number of collisions in a round. Combining the results in Tables V and III, we can calculate the average number of time slots

Table V. Average number of collisions ($COL(n, m)$) in an initialization (or leader election) round.

$n\backslash m$	8	16	32	64	128	256
20	5.8061	5.5856	3.9472	2.354	1.2863	0.6724
40	7.738	11.5	11.273	8.1169	4.9	2.6958
60	7.9743	14.316	17.918	15.272	10.138	5.8626
80	7.9977	15.414	22.9	22.645	16.451	9.9894
100	7.9998	15.805	26.311	29.594	23.418	14.913

that have been chosen to transmit message (denoted as $NT(n, m)$) by the following forms: $NT(n, m) = NS(n, m) + COL(n, m)$, where $NS(n, m)$ is the average number of successful broadcasts in a round with m slots.

By applying the results in Table VI, we can calculate the average length of each time slot. Assume that there are n hosts in the MANET and the value of CW is set as $m - 1$. The average length is $T_{\text{avg}}(n, m) = NT(n, m) \times (DIFS + SLOT) + (m - 1) \times ST/m$. For example, when there are 20 hosts in the MANET, the value of *cw* is set as 15 ($m = 16$), *DIFS* is set as 50 µs, *ST* is set as 20 µs, and *SLOT* is set as 200 µs. The average length in this case is $T_{\text{avg}}(20, 16) = (11.454 \times (50 + 200) + (16 - 1) \times 20)/16 = 197.72$ µs. Table VII shows the average length ($T_{\text{avg}}(n, m)$) of each time slot for different n and m. When the number of hosts is fixed, the larger the size of the contention window, Q13 the smaller is• the average length of each time slot.

Combining the results in Tables II and VII, we can evaluate the average time required for a successful leader election (denoted as $LET(n, m)$) by the following forms: $LET(n, m) = (ES(n, m) - 1) \times T_{\text{avg}}(n, m) + SIFS + SLOT$, where $ES(n, m)$ is the average number of time slots required for a successful leader election. Given the number of hosts (say n), the optimal size of contention window for leader election is $CW_{le}(n) = \{m|LET(n, m) = \min(LET(n, m')),$ where $m' = 2^{3+i}, i = 0, \ldots, 5\}$. Table VIII shows the optimal size of the contention window for a successful leader election with different number of hosts in the MANET. As the number of hosts increases, the optimal size of contention window also increases.

Table VI. Average number of time slots that have been chosen to transmit message in a round with m slots.

$n\backslash m$	8	16	32	64	128	256
20	7.388	11.454	14.888	17.182	18.517	19.239
40	7.957	14.728	22.869	29.76	34.359	37.033
60	7.997	15.648	27.137	38.965	47.911	53.491
80	7.9997	15.903	29.414	45.7	59.504	68.712
100	7.9999	15.973	30.626	50.626	69.42	82.79

Table VII. Average length(μs) of each time slot for different n and m.

$n\backslash m$	8	16	32	64	128	256
20	248.38	197.72	135.69	86.804	56.01	38.71
40	266.15	248.88	198.04	135.94	86.951	56.087
60	267.41	263.24	231.38	171.9	113.42	72.159
80	267.49	267.23	249.17	198.2	136.06	87.024
100	267.5	268.33	258.64	217.45	155.43	100.77

In a similar method, combining the results in Tables III and VII, we can evaluate the average time required for a host to get its own ID (denoted as $GIDT(n, m)$) by the following forms: $GIDT(n, m) = m/NS(n, m) \times T_{\text{avg}}(n, m) + SIFS + SLOT$. Given the number of hosts (say n), the optimal size of the contention window for initialization is $CW_{in}(n) = \{m|GIDT(n, m) = \min(GIDT(n, m')),$ where $m' = 2^{3+i}, i = 0, \ldots, 5\}$. Table IX shows the optimal size of the contention window for a host to get its own ID with different number of hosts in a MANET. According to different number of hosts, the CW of our protocols proposed in Section 3 can be replaced by Tables VIII and IX.

4.3. Our Protocols without the Knowledge of the Number of Hosts

When the number of hosts is not known in advance, we follow the same protocol described in Section 3.1 to elect the leader, except the value of CW. First we set the value of CW to a predetermined value. If the leader cannot be elected in an election round, in the next election round, the size of the contention window will be doubled until $m = 256$.

Table VIII. Optimal size of the contention window for a successful leader election.

n	$1 \sim 2$	$3 \sim 7$	$8 \sim 14$	$15 \sim 30$	$31 \sim 61$	$62 \sim$
m	8	16	32	64	128	256

Table IX. Optimal size of the contention window for a host to get its own ID..

n	$1 \sim 4$	$5 \sim 8$	$9 \sim 16$	$17 \sim 31$	$32 \sim 62$	$63 \sim$
m	8	16	32	64	128	256

The initialization protocol proposed herein is also similar to the one described in Section 3.2, except the value of CW. We first set the value of CW to a predetermined value. After each initialization round, the leader will tell hosts without IDs how many collisions are in the initialization round and how many hosts obtain their IDs, so that these hosts can use an interpolation method to estimate how many hosts are in the MANET and how many hosts remain without ID. In the next initialization round, each host will set the size of the contention window according to the estimated number of hosts that remain without IDs. The procedure will continue until there is no collision in the initialization round and the initialization procedure is considered to be over. When the initialization procedure is over, the leader will realize the current number of hosts in the MANET.

For example, there are 32 hosts that get their IDs and 6 collisions in the first initialization round, where CW is set as 127. According to Table V, each host can use an interpolation method to estimate the number of hosts in the MANET: $n = 40 + (60 - 40) \times 6 - 4.9/10.138 - 4.9 = 44.2$. Each host will estimate that there are 44 hosts in the MANET and 12 hosts remain without IDs after the previous initialization round. Therefore, according to Table IX, each host will set the size of the contention window as 31 ($m = 32$) in the next initialization round. The procedure will continue until there is no collision in the initialization round.

5. Simulation Results

To clarify if our evaluation results provide a good guideline to set the size of CW, we simulate our leader election and initialization protocols by setting the size of CW based on our evaluation results and the standard of IEEE 802.11, respectively. We also compare the performance of our initialization protocols with that of the Nakano–Olariu protocols. Our MAC protocol basically follows the IEEE 802.11 standard [29], but the Nakano–Olariu protocols are simulated in a TDMA-based MANET. In each simulation, the transmission rate is 2 M bits/s, *SIFS* is set as 10 μs, *DIFS* is set as 50 μs, *ST* is set as 20 μs, and *SLOT* is set as 200 μs. The number of hosts in the MANET is tuned from 20 to 100.

Four leader election protocols are simulated here: our leader election protocols based on our evaluation results with the knowledge of the number of hosts (denoted as *HSLE(K)* and the value of CW is set according to Table VIII), without the knowledge of the number of hosts (denoted as *HSLE(U)* and the value of CW is set as 127), based on the IEEE 802.11 standard (denoted as 802.11*LE* and the initial value of CW is set as 7; if the protocols cannot finish within a round, in the next round, each host will double the size of CW until CW = 255), and the Nakano–Olariu's leader election protocol without the knowledge of the number of hosts (denoted as *NOLE(U)*). When the leader cannot be elected in the first few time slots, there is a great opportunity that the *NOLE(U)* protocol will never elect a leader, because the probability that each host contends to be a leader becomes smaller after each failed election time slot. Therefore, we only consider the case that the *NOLE(U)* protocol can elect a leader successfully. Figure 5 shows the average time required for the four leader election protocols. The Nakano–Olariu's leader election protocol performs better than our leader election protocols only when the number of hosts is smaller than 40.

Five different initialization protocols are simulated here: our initialization protocols based on our evaluation results with the knowledge of the number of hosts (denoted as *HSIN(K)* and the value of CW is set according to Table IX), without the knowledge of the number of hosts (denoted as *HSIN(U)* and the value of CW is set as 127), based on the IEEE 802.11 standard (denoted as 802.11*IN*), the Nakano–Olariu's initialization protocols with the knowledge of the number of hosts (denoted as *NOIN(K)*), and without the knowledge of the number of hosts (denoted as *NOIN(U)*). Both the Nakano–Olariu's initialization protocols assume that each host has the collision-detection capability. The performance of the five initialization protocols are presented in Figure 6. The Nakano–Olariu's initialization protocols perform better than our initialization protocols, because the mobile host can detect its own transmission status in the Nakano–Olariu assumption. Therefore, it requires

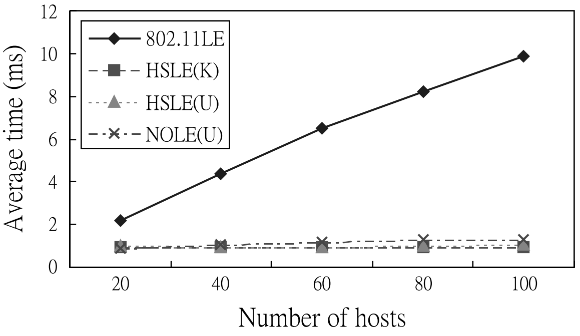

Fig. 5. Average time required for the election protocols.

only one successful broadcast to get a unique ID. However, in the IEEE 802.11-based MANETs, the mobile host requires other hosts to tell its transmission status. Therefore, the *HSIN(K)* and *HSIN(U)* protocols require two successful broadcasts to get a host's unique ID. That is why our initialization protocols will take longer time to finish the job. The performance of our protocols is slightly worse than that of Nakano–Olariu protocols, but our protocols are more practical than the Nakano–Olariu protocols. The performances of our protocols with and without the knowledge of the number of hosts are quite close to each other, which indicates that we have made a good estimation of the number of hosts and set a proper size of the contention window.

Figures 5 and 6 also show that simulating our protocols by setting the size of CW based on our evaluation results performs better than that based on the IEEE 802.11 standard, which indicates that our evaluation results provide a good guideline to set the size of CW.

Figures 7 and 8 show that when based on the Nakano–Olariu assumption, the performance of our leader election and initialization protocols is better than that of the Nakano–Olariu protocols. Since the contention window in our protocols is set properly, our protocols can avoid unnecessary collisions and transmissions, and thus perform better than the Nakano–Olariu protocols.

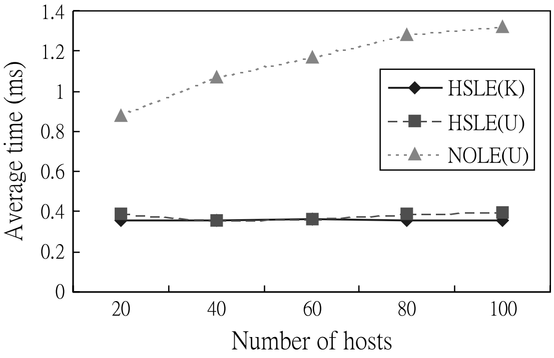

Fig. 7. Average time required for the Nakano–Olariu's and our leader election protocols based on the Nakano–Olariu assumption.

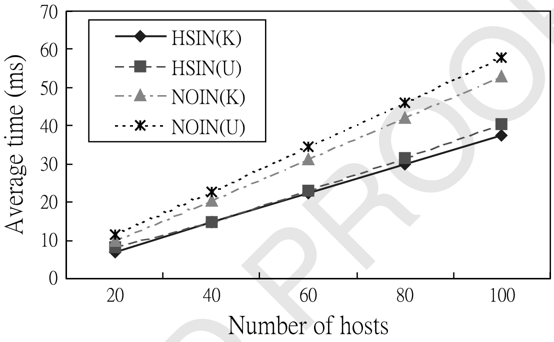

Fig. 8. Average time required for the Nakano–Olariu's and our initialization protocols based on the Nakano–Olariu assumption.

6. Conclusions

In this paper, we have proposed two leader election protocols and initialization protocols for IEEE 802.11-based single-hop MANETs. As we know, no• initialization protocol for IEEE 802.11-based single-hop MANETs has been proposed before. We also proposed several efficient evaluation algorithms, whose time complexities (polynomial) are much lower than the brute force approach (exponential), to evaluate the performance of our protocols. The numerical evaluation results can also be used as a guideline to set the initial value of CW in any IEEE 802.11-based MANET as long as the number of potential contenders is known in advance. Simulation results show that simulating our protocols by setting the size of CW based on our evaluation results performs much better than that based on the IEEE 802.11 standard, which indicates that the evaluation results provide a good guideline to set the size of CW. Besides, our protocols are more practical than the Nakano–Olariu protocols. With a little modification, our protocols can be easily implemented in the IEEE 802.11-based WaveLAN cards.

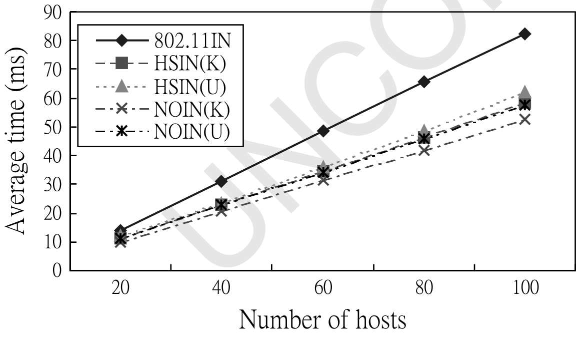

Fig. 6. Average time required for the initialization protocols.

Acknowledgments

This paper was supported by the MOE Program for promoting Academic Excellent of Universities under the grant number A-91-H-FA07-1-4

References

1. Lipton RJ, Park A. The processor identity problem. *Information Processing Letters* 1990●; 91–94. Q15
2. Olariu S, Wen Z. Optimal parallel initialization algorithms for a class of priority queues. *IEEE Transactions on Parallel and Distributed Systems* 1991; **2**: 423–429.
3. Singh G. Efficient distributed algorithms for leader election in complete networks. In *11th International Conference on Distributed Computing Systems*, 1991; pp. 472–479.
4. Nakano K. Optimal initializing algorithms for a reconfigurable mesh. *Journal of Parallel and Distributed Computing* 1995; **24**: 218–223.
5. Dolev S, Israeli A, Moran S. Uniform dynamic self-stabilizing leader election. *IEEE Transactions on Parallel and Distributed Systems* 1997; **8**: 424–440.
6. Olariu S, Stojmenovic I, Zomaya AY. On the dynamic initialization of parallel computers. In *11th International Parallel Processing Symposium*, 1997; pp. 679–683.
7. Francis P, Saxena S. Optimal distributed leader election algorithm for synchronous complete network. *IEEE Region 10 International Conference on Global Connectivity in Energy, Computer, Communication and Control* 1998; **1**: 86–88.
8. Park S-H, Kim Y, Hwang JS. An efficient algorithm for leader-election in synchronous distributed systems. *Proceedings of the IEEE Region 10 Conference* 1999; **2**: 1091–1094.
9. Yamashita M, Kameda T. Leader election problem on networks in which processor identity numbers are not distinct. *IEEE Transactions on Parallel and Distributed Systems* 1999; **10**: 878–887.
10. Stoller SD. Leader election in asynchronous distributed systems. *IEEE Transactions on Computers* 2000; **49**: 283, 284.
11. Shor J, Robertazzi TG. Traffic sensitive algorithms for generating self-organizing network schedules. In *IEEE Military Communications Conference*, 1990; pp. 840–844.
12. El-Ruby M, Kenevan J, Carlson R, Khalil K. A linear algorithm for election in ring configuration networks. *Proceedings of the Twenty-Fourth Annual Hawaii International Conference on System Sciences* 1991; **1**: 117–123.
13. Gerla M, Tsai JTC. Multicluster, mobile, multimedia radio network. *Wireless Networks* 1995; **1**: 255–265.
14. Chang C-H, Tu K, Chen K-C. Efficient end-to-end authentication protocols for mobile networks. *IEEE International Symposium on Personal, Indoor and Mobile Radio Communications* 1996; **3**: 1252–1256.
15. Ikram N. Cryptographic identification of users over network. *IEEE Military Communications Conference* 2001; **1**: 59–63.
16. Sheu JP, Liu CH, Wu SL, Tseng YC. A priority mac protocol to support real-time multimedia traffic in ad hoc networks. *European Wireless* 2002●; 149–155. Q16
17. Kuri J, Kasera SK. Reliable multicast in multi-access wireless LANs. In *IEEE INFOCOM*, 1999; pp. 760–767.
18. Chao HC, Chang SW, Chen JL. Throughput improvements using the random leader technique for the reliable multicast wireless LANs. In *ICN*, 2001.
19. Malpani N, Welch JL, Vaidya N. Leader election algorithms for mobile ad hoc networks. In *ACM DIAL M Workshop*, 2000; pp. 96–103.
20. Nakano K, Olariu S. Randomized initialization protocols for ad hoc network. *IEEE Transactions on Parallel and Distributed Systems* 2000; **11**: 749–759.
21. Nakano K, Olariu S. Energy-efficient initialization protocols for single-hop radio networks with no collision detection. *IEEE Transactions on Parallel and Distributed Systems* 2000; **11**: 851–863.
22. Bhuvaneswaran RS, Bordim JL, Cui J, Ishii N, Nakano K. An energy-efficient initialization protocol for wireless sensor networks. In *International Conference on Parallel Processing Workshops*, 2001; pp. 423–428.
23. Bordim JL, Cui J, Ishii N, Nakano K. Doubly-logarithmic energy-efficient initialization protocols for single-hop radio networks. In *Proceedings International Parallel and Distributed Processing Symposium*, 2002; pp. 145–152.
24. Micic A, Stojmenovic I. A hybrid randomized initialization protocol for TDMA in single-hop wireless networks. In *Proceedings International Parallel and Distributed Processing Symposium*, 2002; pp. 147–154.
25. Nakano K, Olariu S. Uniform leader election protocols for radio networks. *IEEE Transactions on Parallel and Distributed Systems* 2002; **13**: 516–526.
26. Nakano K, Olariu S. A survey on leader election protocols for radio networks. In *Proceedings. International Symposium on Parallel Architectures, Algorithms and Networks*, 2002; pp. 71–76.
27. Park VD, Corson MS. A highly adaptive distributed routing algorithm for mobile wireless networks. In *IEEE INFOCOM*, 1997; pp. 7–11.
28. Kalinowski R, Latteux M, Simplot D. An Adaptive Anti-Collision Protocol for Smart Labels. Technical Report LIFL University Lille, 2001.
29. LAN99 LAN MAN Standards Committee of the IEEE Computer Society. IEEE Std 802.11–1999, Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. IEEE, 1999.

Authors' Biographies

Chih-Shun Hsu received his B.S. degree in computer education from National Taiwan Normal University, Taiwan, Republic of China, in 1990, and the M.S. degree in computer science from National Taiwan University, Taiwan, Republic of China, in 1992. He joined the faculty of the Department of Information Management, Nanya Institute of Technology, Taiwan, Republic of China, as an instructor in 1996. He is currently a Ph.D. candidate of the Department of Computer Science and Information Engineering, National Central University, Taiwan, Republic of China. His current research interests include wireless communications and mobile computing.

Jang-Ping Sheu received the B.S. degree in computer science from Tamkang University, Taiwan, Republic of China, in 1981, and the M.S. and Ph.D. degrees in computer science from the National Tsing Hua University, Taiwan, Republic of China, in 1983 and 1987, respectively.

He joined the faculty of the Department of Electrical Engineering, National Central University, Taiwan, Republic of China, as an associate professor in 1987. He is currently a professor of the Department of Computer Science and Information Engineering, National Central University. He was a Chair of Department of Computer Science and Information Engineering, National Central University, from August 1997 to July 1999. He was a visiting professor at the

Department of Electrical and Computer Engineering, University of California, Irvine, from July 1999 to April 2000. His current research interests include wireless communications, mobile computing, parallel processing, and distributed computing systems.

He was an associate editor of Journal of the Chinese Institute of Electrical Engineering, from August 1, 1996 to July 31, 2000. He was an associate editor of Journal of Information Science and Engineering from August 1, 1996 to July 31, 2002. He is an associate editor of Journal of the Chinese Institute of Engineers. He was a guest editor of Special Issue for Wireless Communications and Mobile Computing Journal. He was a Program Chair of IEEE ICPADS'2002. He received the Distinguished Research Awards of the National Science Council of the Republic of China in 1993–1994, 1995–1996, and 1997–1998.

Dr Sheu is a senior member of the IEEE, a member of the ACM and Phi Tau Phi Society.

QUERIES TO BE ANSWERED BY AUTHOR (SEE MARGINAL MARKS)

IMPORTANT NOTE: Please mark your corrections and answers to these queries directly onto the proof at the relevant place. Do NOT mark your corrections on this query sheet.

Query No.	Query
Q1	We have replaced ‘id’ with ‘ID’ throughout in this article. Please clarify if this is acceptable.
Q2	We have rephrased this sentence. Please clarify if it retains the intended meaning.
Q3	Please clarify if ‘Excellent’ should be changed to ‘Excellence’.
Q4	We have rephrased ‘As our best knowledge’ as ‘To the best of our knowledge’ in this sentence. Please clarify if it retains the intended meaning.
Q5	Please clarify if ‘TORA’ has to be spelt out at the first instance. If so, please provide the expansion.
Q6	Please clarify if ‘TDMA’ should be spelt out as ‘Time-division Multiple access’. If not, please provide the expansion.
Q7	May we replace ‘works’ with ‘protocols’?
Q8	May we replace ‘Ack’ with ‘ACK’ throughout in this article?
Q9	We have rephrased ‘no matter’ as ‘irrespective of whether’ in this sentence. Please clarify if the sentence retains the intended meaning.
Q10	May we rephrase ‘...and complete the initialization...’ as ‘...and this completes the initialization...’?
Q11	We have combined this sentence with the previous one. Please clarify if this is fine.
Q12	We have rephrased this sentence. Please clarify if we have retained the intended meaning.
Q13	We have rephrased this sentence. Please clarify if we have retained the intended meaning.
Q14	We have deleted the word ‘that’ from this sentence. Please clarify if this is fine.
Q15	Please provide the volume number for this reference.
Q16	Please provide the volume number for this reference.